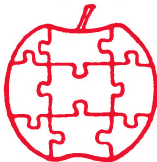


Apple

\$1.80



Assembly

Line

Volume 5 -- Issue 3

December, 1984

In This Issue...

18-Digit Arithmetic, Part 8.	2
More Details on Using 65C02's in Older Apples.	15
"Inside the Apple //e", a Review	16
Correction re MVN and MVP in 65802	18
Strange Way to Divide by 7	19
Sly Hex Conversion	21
Remembering When	23
Generating Tables for Faster Hi-Res.	24
Blankenship's BASIC.	26
Solution to Overlapping DOS Patches.	27

New Source for 65802's

I talked to Constantine Geromnimon at Alliance Computers this morning. His company has ordered hundreds of 65802's, and offers them to you at \$49.95 each. They expect their next shipment to come in around the middle of January, so now is the time to order. Call them at (718) 672-0684, or write to P. O. Box 408, Corona, NY 11368.

Tom Weishaar Writes Again!

If you are among the throng who mourn the passing of Softalk, and particularly of the many informative columns such as DOSTalk by Tom Weishaar, you will be as glad as I am that Tom has started publishing his own monthly newsletter.

Called "Open-Apple", you can subscribe for \$24. In an unprecedented move toward international goodwill and the wholesome exchange of information, Tom has set the price the same for everyone, everywhere. We promptly sent him a check. If you love your Apple, do likewise. Send to Open-Apple, 10026 Roe, Overland Park, Kansas 66207. If you are cautious, send no money; Tom will bill you with the first issue, and you can cancel if you lose interest.

Someone pointed out last week that this series is getting a little long. Well, we are nearing the end. What we are doing is probably unprecedented in the industry: listing the source code and explaining it for a large commercially valuable software product. It takes time and space to break precedents.

This month's installment completes the normal set of math functions, with sine, cosine, and arc tangent. We even slipped in a simple form of the tangent function. Still to come are the formatted INPUT and PRINT routines.

Some Elementary Info:

Trigonometry is a frightening word. (If it doesn't scare you, skip ahead several paragraphs.) The "-ometry" refers to measurement, but what is a "trigon". Believe it or not, "trigon" is another name for a triangle. Trigon means three sides, and figures with three sides just happen to also have three angles. "Trig" (a nice nickname) is a branch of mathematics dealing with triangles, without which we could not fly to the moon, draw a map, or build bridges. Strangely enough, much of electronics also uses trig functions ... are electrons triangular?

When I took trig in high school, long before the day of personal calculators, we used trig tables. (These were not articles of furniture made in the local woodshop, but rather long lists of strange numbers printed and bound into books.) The tables contained values for various ratios of the sides of a triangle having one 90-degree angle. Now we use calculators or computers, but obviously the trig tables would not fit in them. Instead, approximation formulas are used.

In high school, we talked about six different ratios: sine, cosine, tangent, cotangent, secant, and cosecant. When it is all boiled down, we really only need the sine; all the rest are derivable from those. The sine function gives a number for any angle. We frequently need to be able to go from a trig value back to an angle, and the most useful function for that is called the inverse tangent, or arctangent.

Even though I have been talking about triangles, trig functions are even more related to circles. We compute functions of the angle between any two radii, like the hands on an old-fashioned, pre-digital wrist watch. When we start talking about circles, we get into radians vs. degrees.

Just as scientists like logarithms to the base e (rather than 10), they also like trig functions based on angles expressed in radians, rather than degrees. Degrees were invented back in Babylon, I understand, and are nice and clean: 360 make a complete circle. Radians are not clean: 360 degrees is two-times-pi radians. Nevertheless, many physical and electronic formulas simplify when angles are expressed in radians. Consequently, calculators and computer languages usually expect your angles to be expressed in radians. Some

S-C Macro Assembler Version 1.0.....\$80
 S-C Macro Assembler Version 2.0.....\$100
 Version 2.0 Update.....\$20
 Source Code for Version 1.1 (on two disk sides).....\$100
 Full Screen Editor for S-C Macro (with complete source code).....\$49
 S-C Cross Reference Utility (without source code).....\$20
 S-C Cross Reference Utility (with complete source code).....\$50
 DISASM Dis-Assembler (RAK-Ware).....\$30
 Source Code for DISASM.....additional \$30

S-C Word Processor (with complete source code).....\$50
 Double Precision Floating Point for Applesoft (with source code).....\$50
 S-C Documentor (complete commented source code of Applesoft ROMs).....\$50
 Source Code of //e CX & F8 ROMs on disk.....\$15

(All source code is formatted for S-C Macro Assembler Version 1.1. Other assemblers require some effort to convert file type and edit directives.)

AAL Quarterly Disks.....each \$15
 Each disk contains all the source code from three issues of "Apple Assembly Line", to save you lots of typing and testing time.
 QD#1: Oct-Dec 1980 QD#2: Jan-Mar 1981 QD#3: Apr-Jun 1981
 QD#4: Jul-Sep 1981 QD#5: Oct-Dec 1981 QD#6: Jan-Mar 1982
 QD#7: Apr-Jun 1982 QD#8: Jul-Sep 1982 QD#9: Oct-Dec 1982
 QD#10: Jan-Mar 1983 QD#11: Apr-Jun 1983 QD#12: Jul-Sep 1983
 QD#13: Oct-Dec 1983 QD#14: Jan-Mar 1984 QD#15: Apr-Jun 1984
 QD#16: Jul-Sep 1984 QD#17: Oct-Dec 1984

AWIIe Toolkit (Don Lancaster, Synergetics).....\$39
 Quick-Trace (Anthro-Digital).....CLOSEOUT SPECIAL!..(reg. \$50) ~~\$45~~ \$35
 Visible Computer: 6502 (Software Masters).....(reg. \$50) \$45
 ES-CAPE: Extended S-C Applesoft Program Editor.....\$60
 "Bag of Tricks", Worth & Lechner, with diskette.....(\$39.95) \$36

Blank Diskettes (Verbatim).....\$2.25 each, or package of 20 for \$40
 (Premium quality, single-sided, double density, with hub rings)
 Vinyl disk pages, 6"x8.5", hold two disks each.....10 for \$6
 Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each
 or \$25 per 100

These are cardboard folders designed to fit into 6"x9" Envelopes.
 Envelopes for Diskette Mailers.....6 cents each
 quikLoader EPROM System (SCRG).....(\$179) \$170
 D Manual Controller (SCRG).....(\$90) \$85
 Switch-a-Slot (SCRG).....(\$190) \$175
 Extend-a-Slot (SCRG).....(\$35) \$32
 PROMgrammer (SCRG).....(\$149.50) \$140

Books, Books, Books.....compare our discount prices!

"Inside the Apple //e", Little.....(\$19.95) \$18
 "Apple II+/IIe Troubleshooting & Repair Guide", Brenner.....(\$19.95) \$18
 "Apple II Circuit Description", Gayler.....(\$22.95) \$21
 "Understanding the Apple II", Sather.....(\$22.95) \$21
 "Enhancing Your Apple II, vol. 1", Lancaster.....(\$15.95) \$15
 Second edition, with //e information.
 "Assembly Cookbook for the Apple II/IIe", Lancaster.....(\$21.95) \$20
 "Incredible Secret Money Machine", Lancaster.....(\$7.95) \$7
 "Beneath Apple DOS", Worth & Lechner.....(\$19.95) \$18
 "Beneath Apple ProDOS", Worth & Lechner.....(\$19.95) \$18
 "What's Where in the Apple", Second Edition.....(\$19.95) \$19
 "6502 Assembly Language Programming", Leventhal.....(\$18.95) \$18
 "6502 Subroutines", Leventhal.....(\$18.95) \$18
 "Real Time Programming -- Neglected Topics", Foster.....(\$9.95) \$9
 "Microcomputer Graphics", Myers.....(\$12.95) \$12

Add \$1.50 per book for US shipping. Foreign orders add postage needed.

Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
 *** (214) 324-2050 ***
 *** We accept Master Card, VISA and American Express ***

allow both options. Applesoft expects radians, and so do my DP18 programs.

We commonly think of an angle as being somewhere between 0 and 360 degrees, or the equivalent range in radians. However, angles can actually be any number, from -infinity to +infinity. The numbers beyond one complete circle are valid, but they don't buy much. If you stand in one place and spin around 1445 degrees ($4 \times 360 + 5$) you will end up pointing the same direction as if you merely swiveled 5 degrees. Therefore the first step in a sine function calculation involves subtracting out all the multiples of a full circle from the angle.

The arctangent function could return an infinite number of answers, but that is impractical. We will return only the principal value, which is the one closest to 0. All others are that value plus or minus any number of full circles. In DP18 the ATN function may have one or two arguments. If you only have one argument, the result will be an angle between $-\pi/2$ and $+\pi/2$. If you specify two arguments, a value between $-\pi$ and $+\pi$ will be returned.

The Nitty-Gritty:

Enough of this preliminary stuff, let's get into the code. In the listing which follows, you will find entries for four functions: SIN, COS, TAN, and ATN.

Perhaps the easiest is the TAN function, at lines 2530-2630. Since $\tan = \sin/\cos$, that is all this code does. We lose a little speed and possibly some precision with this simplistic solution, but the TAN function is relatively rarely called.

Next in difficulty is the COS function, lines 1630-1710. Since $\cos(-x) = \cos(x)$, we start by making the sign positive (lines 1690-1700. Since $\cos(x) = \sin(x + \pi/2)$, we add $\pi/2$ and fall into the SIN function. Simple, but effective.

The SIN function gets more interesting. For very very small angles, within the precision of 20 digits, $\sin(x) = x$. Lines 1780-1810 check for exponents below -10; all angles smaller than 10^{-10} are small enough that $\sin(x) = x$.

Next we take advantage of the fact that $\sin(-x) = -\sin(x)$, at lines 1820-1860. We remember the sign by shoving it on the stack, and force the sign of x positive.

Lines 1870-1950 get the principal angle. I divide x by 2π , and throw away the integral part. The fractional part that remains is a fraction of a full circle, a value between 0 and .999999...9 (not radians, and not degrees either). Note that if x was extremely large there will be no fractional part, and the remainder will be zero. Some SIN function calculators give an error message when this happens, but I chose to let it ride.

Lines 1960-2000 multiply the circle-fraction by four. This gives a number between 0 and 3.99999...9, which I will refer to later as the "circle fraction times four", or c-f-t-f. The

integer part is effectively a quadrant number, and the fractional part a fraction within the quadrant:

1	0
2	3

Lines 2010-2030 determine if the angle is in the first (0) quadrant. If so, no folding need be done.

Lines 2040-2070 determine if the angle is in the second (1) quadrant. If so, we skip ahead to apply the fact that $\sin(\pi/2 + x) = \sin(\pi/2 - x)$.

Lines 2080-2160 are executed if the angle is in the 3rd or 4th quadrants (integral part is 2 or 3). Here I apply the fact that $\sin(\pi+x)=-\sin(x)$. I pull the saved sign off the stack, complement it, and shove it back on (lines 2090-2110). Then I subtract 2 from the c-f-t-f, yielding a number between 0 and 1.99999...9. We have folded the third and fourth quadrants over the first and second quadrants. Next lines 2170-2190 determine if the result was in the first quadrant or not.

Lines 2200-2240 fold a second quadrant number into the first quadrant, by applying the fact that $\sin(\pi/2+x) = \sin(\pi/2-x)$. Subtracting the c-f-t-f from 2 flips us into the first quadrant.

Lines 2260-2270 pull the sign off the stack and make it the sign of the angle. Remember that now the angle is a fraction (between 0 and .99999...9) of a quadrant. After all these folding operations, the angle might again be very very small, so lines 2280-2300 check for that possibility. If so, $\sin(x)=x$, but that is only true when x is in radians. Lines 2490-2520 convert the quadrant-fraction to radians by multiplying by $\pi/2$, and exits.

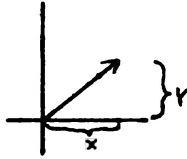
Lines 2310-2470 handle larger angles by computing $x \cdot P/Q$, where P and Q are polynomials in x^2 . The constants for P and Q are given in lines 1420-1550, and come from the Hart book. [I should mention here that I wrote those constants with pretty periods separating groups of five digits. This will not assemble in some older versions of the S-C Macro Assembler. If you get a syntax error, just leave out the periods.]

Turning the Tables:

ATN is hardest to compute. First we have to deal with the two variants of calls, having one or two arguments. While all the previous function programs were called with the argument already in DAC, DP.ATN is called immediately after parsing the ATN-token. Lines 2960-3070 parse and process the following parentheses and whatever is between them.

Lines 2960-2970 require an opening parenthesis. Line 3070 requires the closing parenthesis. In between we expect one expression, or two expressions separated by a comma. If there is only one, we fake a second one (= 1.0).

What are the two arguments? Looking at a cartesian system, with the vector shown below, the arguments are (Y,X). If you call with one argument, it is (Y/X).



By using two separate arguments, rather than just the ratio, we can tell which of the four quadrants the vector was in. DP.ATAN will return a value between $-\pi$ and $+\pi$, depending on the two signs. If you specify only the ratio, DP.ATAN will return a value between 0 and $+\pi$ depending on the sign.

Lines 3120-3160 save the two signs in bits 6 and 7 of UV.SIGN. Way at the end, lines 4100 and following, UV.SIGN determines the final value. If the sign of the denominator (X-vector) was negative, the composite vector is in the 2nd or 3rd quadrant: computing $\pi - \text{angle}$ gives a result between $\pi/2$ and π .

If the numerator (Y-vector) was negative, the composite vector is in the 3rd or 4th quadrant. Flipping the sign gives a result between 0 and $-\pi$.

Lines 3180-3220 check for special cases of $Y=0$ or $X=0$. If the first argument (Y-vector) is zero, the angle is 0 or π depending on the sign of the second argument. If the second argument (X-vector) is zero, the angle is either $+\pi/2$ or $-\pi/2$, depending on the sign of the first argument. What if both arguments are zero? That should produce an error message, but I am overlooking it: I will return an angle of 0 in this case.

If neither argument is zero, some special checks are made to see if the value of the ratio is very small or very large. I check before actually dividing, so the divide routine won't kick out on an overflow error. If the ratio would be greater than 10^{20} , I return a value of $\pi/2$. This is accurate within the precision of DP18. On the other hand, if the ratio is smaller than 10^{-63} I return 0. If neither extreme is true, I go ahead and divide to get the actual ratio. Then I check for an extremely small ratio, in which case $\text{atan}(x)=x$.

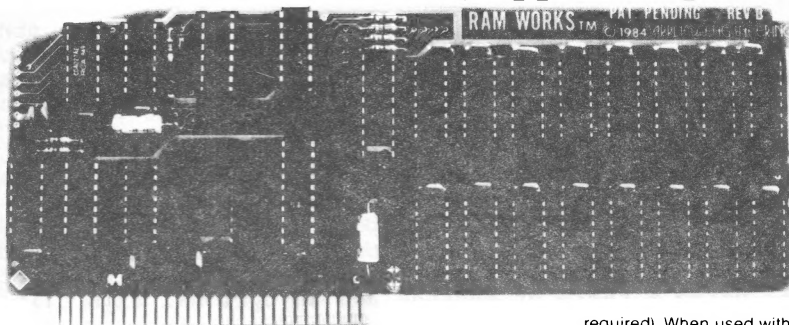
If we find our way down to line 3390, the ratio is between 10^{-10} and 10^{20} . That is still too large a range for comfort, so we apply the fact that $\text{atan}(1/x) = \text{atan}(\pi/2 - x)$. If the ratio of Y/X is greater than 1.0, then we take the reciprocal and remember that we did so. This in effect folds the range at $\pi/4$. The resulting argument range is between 10^{-10} and 1. The variable N holds either 0 or 2 as a flag: 0 if we were already under 1, 2 if we formed the reciprocal.

The shape of the curve of the arctangent function between 0 and 1 (an angle between 0 and $\pi/4$) is deceptive. It looks nice

RAMWORKS™

800K Apple Works Desk Top, 450K Visicalc, 1 Meg Solid State Disk

Double Hi-Res, RGB, Totally Apple Compatible!



RAMWORKS™—A card that plugs into the Apple IIe auxiliary slot and functions EXACTLY like Apple's extended 80 column card (in fact, a 128K RAMWORKS™ actually costs less than Apple's 64K extended card) but with RAMWORKS™ you get more memory, 80 column text, a 3-year warranty and most importantly, room to grow without using more slots. A design so advanced there's a patent pending on it. If you have a IIc or an IBM, we suggest you do what everybody does, trade it in on a IIe.

RAMWORKS™ can be purchased in a wide range of sizes and is user upgradeable using either 64K RAMS or the new 256K RAMS. In fact, RAMWORKS™ is the only auxiliary slot card on the market that will allow the new 256K RAMS to be used. If you already have an extended 80 column card, no problem. Just unplug the 64K RAMS and plug them into the RAMWORKS™ for an additional 64K. A RGB option is also available, you can order it with your RAMWORKS™ card or add it on at a later date.

RAMWORKS™ saves you time, money, slots and hassle. You'll have additional memory NOW and in the future.

Ramworks™

64K Installed	\$ 179
128K Installed	\$ 249
256K Installed	\$ 399
512K Installed	\$ 649
1 MEG Installed	\$1199
RGB Option.....	\$ 129

(May be added later)

LOW COST SOFTWARE OPTIONS

Ram Drive IIe

Ram Drive IIe will give you a high speed solid state disk drive. The Ram Drive IIe software features audio-visual access indicators, easy setup for turnkey operation, and easy menu driven documentation. The program can be modified and is copyable. If you have a 64K RAMWORKS™, Ram Drive IIe will act as half a disk drive. If you have a 128K or larger RAMWORKS™, Ram Drive IIe will act as a full disk drive. Either way, your programs will load and save over 20 times faster. Ram Drive IIe is compatible with APPLESOFT, PRO DOS, DOS 3.3, and PASCAL. The disk also includes a high speed RAMdisk copying program. Ram Drive is an entire disk drive only 20 times faster. And no whirring, clicking and waiting!

PRICE \$29

CP/M Ram Drive IIe

CP/M Ram Drive IIe is just like the Ram Drive IIe above, only for CP/M. CP/M Ram Drive IIe runs on any Z-80 card that runs standard CP/M i.e. Applied Engineering Z-80 Plus or Microsoft Soft Card. CP/M Ram Drive will dramatically speed up the operation of most CP/M software because CP/M normally goes to disk fairly often. Fast acting software like dBase II, Wordstar and Turbo Pascal becomes virtually instantaneous when used with CP/M Ram Drive.

PRICE \$29

VC IIe Expander

VC IIe expander gives owners of Visicalc IIe and Advanced Visicalc IIe increased storage. When used with VC IIe you'll get 141K work-space (128K RAMWORKS™ or larger

required). When used with Advanced VC IIe you'll get 131K with a 128K RAMWORKS™, 250K with a 256K RAMWORKS™ and 450K with a 512K RAMWORKS™ **PRICE \$29**

Apple Works Expand

Although Apple Works is compatible with all sizes of RAMWORKS™, Apple Works only "sees" its first 64K bank giving you a 55K desktop. Our Apple Works expand program will make a modification to Apple Works that simply lets it know you've got more memory, giving you 101K work space. **PRICE \$29**

Super Apple Works Expand

This souped-up version of Apple Works expand doesn't stop at a 101K desk top, in fact Super Apple Works Expand figures out how much memory your RAMWORKS™ has to give Apple Works the following desk-top sizes:

RAMWORKS™	APPLEWORKS DESKTOP
128K	101K
256K	200K
512K	400K
1 MEG	800K

PRICE \$39

AppleWorks can also be put in the RAMWORKS™ card to eliminate disk access, thereby dramatically speeding up the program.



APPLIED ENGINEERING

Send Check or Money Order to
Applied Engineering
PO Box 798 Carrollton TX 75006
Call (214) 482-2027

Ram to 11 am 7 days a week MasterCard Visa & C.O.D. Welcome No extra charge for credit cards Texas residents add 5.4% sales tax Add \$10.00 if outside U.S.A.

and easy, but a polynomial over that range with 20 digits of precision is much too long. We can easily reduce the range still further by applying another identity. If the reduced argument is now already below $\tan(\pi/12)$, fine. If not, calculating $(x*\text{sqr}(3)-1) / (\text{sqr}(3)+x)$ will bring it into that range. If we have to apply that formula, N will be incremented (making it 1 or 3).

The curve between 0 and $\tan(\pi/12)$ looks almost like a straight line to the naked eye, but it really is far from straight. It takes a ratio of the form P/xQ where P and Q are polynomials in x^2 . The coefficients are given in lines 2650-2770, again from Hart. The ratio is computed in lines 3800-3960.

Lines 3970-4080 start the unfolding process. The variable N is either 0, 1, 2, or 3 by this time. If N is 0, no folding was done. If N is 1, only folding above $\pi/12$ was done. If N is 2, only folding above $\pi/4$ was done. If N is 3, both folds were done. These lines convert the angle back to the correct value, using a table of addends and an optional sign flip:

N	unfolding formula
0	none
1	$\pi/6 + x$
2	$\pi/2 - x$
3	$\pi/2 - (\pi/6 + x) = \pi/3 - x$

That's it! We already discussed the code beyond line 4100, which figures out which quadrant the angle is in.

Any questions?

```

1000 *SAVE S.DP18 TRIG
1010 *-----
B1- 1020 AS.CHRGET .EQ $B1
B7- 1030 AS.CHRGOT .EQ $B7
DEBB- 1040 AS.CHKCLS .EQ $DEBB
DEBB- 1050 AS.CHKOPN .EQ $DEBB
1060 *-----
FFFF- 1070 POLY.1 .EQ $FFFF
FFFF- 1080 POLY.N .EQ $FFFF
FFFF- 1090 DADD .EQ $FFFF
FFFF- 1100 DSUB .EQ $FFFF
FFFF- 1110 DMULT .EQ $FFFF
FFFF- 1120 DDIV .EQ $FFFF
FFFF- 1130 DP.INT .EQ $FFFF
FFFF- 1140 DP.EXP .EQ $FFFF
FFFF- 1150 DP.TRUE .EQ $FFFF
FFFF- 1160 DP.FALSE .EQ $FFFF
FFFF- 1170 MOVE.DAC.ARG .EQ $FFFF
FFFF- 1180 MOVE.YA.ARG.1 .EQ $FFFF
FFFF- 1190 MOVE.YA.DAC.1 .EQ $FFFF
FFFF- 1200 SWAP.ARG.DAC .EQ $FFFF
FFFF- 1210 MOVE.DAC.TEMP1 .EQ $FFFF
FFFF- 1220 MOVE.DAC.TEMP2 .EQ $FFFF
FFFF- 1230 MOVE.DAC.TEMP3 .EQ $FFFF
FFFF- 1240 MOVE.TEMP1.DAC .EQ $FFFF
FFFF- 1250 MOVE.TEMP1.ARG .EQ $FFFF
FFFF- 1260 MOVE.TEMP2.ARG .EQ $FFFF
FFFF- 1270 MOVE.TEMP3.ARG .EQ $FFFF
FFFF- 1280 PUSH.DAC.STACK .EQ $FFFF
FFFF- 1290 POP.STACK.ARG .EQ $FFFF

```


0800-	1300	*-----		
0801-	1310	DAC.EXPONENT	.BS 1	
080B-	1320	DAC.HI	.BS 10	
	1330	DAC.SIGN	.BS 1	
	1340	*-----		
080C-	1350	ARG.EXPONENT	.BS 1	
080D-	1360	ARG.HI	.BS 10	
0817-	1370	ARG.SIGN	.BS 1	
	1380	*-----		
0818-	1390	N	.BS 1	
0819-	1400	UV.SIGN	.BS 1	
	1410	*-----		
081A-	1420	P.SIN	.EQ *	
06-	1430	P.SIN.N	.EQ 6	$P6 * X^6 + P5 * X^5 + \dots + P1 * X + P0$
081A-	3C	50	31	
081D-	26	38	84	
0820-	64	66	41	
0823-	28	45		1440
0825-	BE	82	81	.HS 3C.50312.63884.64664.12845 P6
0828-	80	80	39	
082B-	29	57	73	
082E-	91	10		1450
0830-	40	62	91	.HS BE.82818.08039.29577.39110 P5
0833-	96	34	90	
0836-	93	11	35	
0839-	52	30		1460
083B-	C2	25	64	.HS 40.62919.63490.93113.55230 P4
083E-	24	40	36	
0841-	60	33	85	
0844-	70	70		1470
0846-	43	53	89	.HS C2.25642.44036.60338.57070 P3
0849-	26	40	53	
084C-	57	78	87	
084F-	62	89		1480
0851-	C4	49	32	.HS 43.53892.64053.57788.76289 P2
0854-	66	74	70	
0857-	47	15	23	
085A-	66	77		1490
085C-	45	12	59	.HS C4.49326.67470.47152.36677 P1
085F-	61	63	80	
0862-	91	36	54	
0865-	18	16		1500
				.HS 45.12596.16380.91365.41816 P0
0867-	1510	*-----		
02-	1520	Q.SIN	.EQ *	
0867-	43	15	74	
086A-	34	33	16	
086D-	33	19	41	
0870-	39	35		1540
0872-	44	80	18	.HS 43.15743.43316.33194.13935 Q1
0875-	96	69	36	
0878-	87	72	71	
087B-	57	87		1550
				.HS 44.80189.66936.87727.15787 Q0
	1560	*-----		
087D-	41	10	00	
0880-	00	00	00	
0883-	00	00	00	
0886-	00	00		1570
0888-	41	20	00	CON.ONE .HS 41.10000.00000.00000.00000
088B-	00	00	00	
088E-	00	00	00	
0891-	00	00		1580
0893-	41	62	83	CON.TWO .HS 41.20000.00000.00000.00000
0896-	18	53	07	
0899-	17	95	86	
089C-	47	69		1590
089E-	41	15	70	CON.2PI .HS 41.62831.85307.17958.64769
08A1-	79	63	26	
08A4-	79	48	96	
08A7-	61	92		1600
08A9-	41	31	41	CON.PI.2 .HS 41.15707.96326.79489.66192
08AC-	59	26	53	
08AF-	58	97	93	
08B2-	23	85		1610
08B4-	40	15	91	CON.PI .HS 41.31415.92653.58979.32385
08B7-	54	94	30	
08BA-	91	89	53	
08BD-	35	77		1620
				CON.1..2PI .HS 40.15915.49430.91895.33577 1/2PI

```

1630 *-----
1640 *      COS (DAC)
1650 *-----
08BF- A9 9E 1660 DP.COS LDA #CON.PI.2      PI/2
08C1- A0 08 1670 LDY /CON.PI.2
08C3- 20 FF FF 1680 JSR MOVE.YA.ARG.1  COS(X) = SIN(X+PI/2)
08C6- A9 00 1690 LDA #0      GET ABS(DAC) TO FORCE
08C8- 8D 0B 08 1700 STA DAC.SIGN  ...COS(-X)=COS(X)
08CB- 20 FF FF 1710 JSR DADD
1720 *-----
1730 *      SIN (DAC)
1740 *      #3371
1750 *-----
1760 DP.SIN
1770 *---IF X VERY SMALL...-----
08CE- AD 00 08 1780 LDA DAC.EXPONENT
08D1- C9 36 1790 CMP #40-10
08D3- B0 01 1800 BCS .1      NOT VERY SMALL
08D5- 60 1810 RTS      VERY SMALL, SIN(X)=X
1820 *---ADJUST FOR SIGN OF X-----
08D6- AD 0B 08 1830 .1 LDA DAC.SIGN  SIN(-X) = - SIN(X)
08D9- 48 1840 PHA      ...SO SAVE SIGN OF X
08DA- A9 00 1850 LDA #0      ...AND MAKE X POSITIVE
08DC- 8D 0B 08 1860 STA DAC.SIGN
1870 *---X*(1/2PI)-----
08DF- A9 B4 1880 LDA #CON.1..2PI
08E1- A0 08 1890 LDY /CON.1..2PI
08E3- 20 FF FF 1900 JSR MOVE.YA.ARG.1
08E6- 20 FF FF 1910 JSR DMULT
1920 *---GET FRACTIONAL PART-----
08E9- 20 FF FF 1930 JSR MOVE.DAC.ARG
08EC- 20 FF FF 1940 JSR DP.INT
08EF- 20 FF FF 1950 JSR DSUB
1960 *---FOLD QUADRANTS INTO ONE-----
08F2- 20 FF FF 1970 JSR MOVE.DAC.ARG  MULTIPLY BY FOUR
08F5- 20 FF FF 1980 JSR DADD  BY DOUBLING TWICE
08F8- 20 FF FF 1990 JSR MOVE.DAC.ARG
08FB- 20 FF FF 2000 JSR DADD  0 <= DAC < 4
08FE- AD 00 08 2010 LDA DAC.EXPONENT  IS DAC < 1?
0901- C9 41 2020 CMP #41
0903- 90 29 2030 BCC .4      ...YES, IT IS IN 1ST QUADRANT
2040 *---2ND, 3RD, OR 4TH-----
0905- AD 01 08 2050 LDA DAC.HI
0908- C9 20 2060 CMP #20  IS DAC < 2.0?
090A- 90 18 2070 BCC .3      ...YES, 1ST OR 2ND QUADRANT
2080 *---FOLD 3RD-4TH OVER 1ST-2ND-----
090C- 68 2090 PLA
090D- 49 80 2100 EOR #80
090F- 48 2110 PHA
0910- A9 88 2120 LDA #CON.TWO  FOLD 3RD & 4TH OVER 1ST & 2ND
0912- A0 08 2130 LDY /CON.TWO
0914- 20 FF FF 2140 JSR MOVE.YA.ARG.1
0917- 20 FF FF 2150 JSR SWAP.ARG.DAC
091A- 20 FF FF 2160 JSR DSUB
091D- AD 00 08 2170 LDA DAC.EXPONENT
0920- C9 41 2180 CMP #41
0922- 90 0A 2190 BCC .4      ...ALREADY IN 1ST
2200 *---FOLD 2ND OVER 1ST-----
0924- A9 88 2210 .3 LDA #CON.TWO  LET X=2-X
0926- A0 08 2220 LDY /CON.TWO
0928- 20 FF FF 2230 JSR MOVE.YA.ARG.1
092B- 20 FF FF 2240 JSR DSUB
2250 *---ANGLE NOW IN 1ST QUADRANT-----
092E- 68 2260 .4 PLA  PUT FINAL SIGN ON X
092F- 8D 0B 08 2270 STA DAC.SIGN
0932- AD 00 08 2280 LDA DAC.EXPONENT  CHECK FOR VERY SMALL
0935- C9 37 2290 CMP #40-9
0937- 90 2D 2300 BCC .5      ...YES, SIN(X)=X*PI/2
0939- 20 FF FF 2310 JSR MOVE.DAC.ARG  PREPARE FOR POLYNOMIALS
093C- 20 FF FF 2320 JSR MOVE.DAC.TEMP1  X IN TEMP1
093F- 20 FF FF 2330 JSR DMULT  X*X IN TEMP2
0942- 20 FF FF 2340 JSR MOVE.DAC.TEMP2
0945- A9 1A 2350 LDA #P.SIN
0947- A0 08 2360 LDY /P.SIN
0949- A2 06 2370 LDX #P.SIN.N
094B- 20 FF FF 2380 JSR POLY.N
094E- 20 FF FF 2390 JSR MOVE.DAC.TEMP3

```

S-C Software Corporation

2331 Gus Thomasson, Suite 125, P.O. Box 280300, Dallas, Texas 75228 (214) 324-2050

S-C Macro Cross Assemblers

The high cost of dedicated microprocessor development systems has forced many technical people to look for alternate methods to develop programs for the various popular microprocessors. Combining the versatile Apple II with the S-C Macro Assembler provides a cost effective and powerful development system. Hobbyists and engineers alike will find the friendly combination the easiest and best way to extend their skills to other microprocessors.

The S-C Macro Cross Assemblers are all identical in operation to the S-C Macro Assembler; only the language assembled is different. They are sold as upgrade packages to the S-C Macro Assembler. The S-C Macro Assembler, complete with 100-page reference manual, costs \$80; once you have it, you may add as many Cross Assemblers as you wish at a nominal price. The following S-C Macro Cross Assembler versions are now available:

Motorola:	6800,1,2/6301	\$32.50	RCA:	1802/1805	\$32.50
	6805	\$32.50			
	6809	\$32.50	Rockwell:	65C02	\$20
	68000	\$50			
			DEC:	LSI-11	\$50
Intel:	8048	\$32.50	General Instruments:		
	8051	\$32.50		GI-1650	\$50
	8085	\$32.50		GI-1670	\$50
Zilog:	Z-80	\$32.50			
	Z-8	\$32.50			

The S-C Macro Assembler family is well known for its ease-of-use and powerful features. Thousands of users in over 30 countries and in every type of industry attest to its speed, dependability, and user-friendliness. There are 20 assembler directives to provide powerful macros, conditional assembly, and flexible data generation. INCLUDE and TARGET FILE capabilities allow source programs to be as large as your disk space. The integrated, co-resident source program editor provides global search and replace, move, and edit. The EDIT command has 15 sub-commands combined with global selection.

Each S-C Assembler diskette contains two complete ready-to-run assemblers: one is for execution in the mother-board RAM; the other executes in a 16K RAM Card. The HELLO program offers menu selection to load the version you desire. The disks may be copied using any standard Apple disk copy program, and copies of the assembler may be BSAVED on your working disks.

S-C Software Corporation has frequently been commended for outstanding support: competent telephone help, a monthly (by subscription) newsletter, continuing enhancements, and excellent upgrade policies.

```

0951- A9 67 2400 LDA #Q.SIN
0953- A0 08 2410 LDY /Q.SIN
0955- A2 02 2420 LDX #Q.SIN.N
0957- 20 FF FF 2430 JSR POLY.1
095A- 20 FF FF 2440 JSR MOVE.TEMP3.ARG
095D- 20 FF FF 2450 JSR DDIV P/Q
0960- 20 FF FF 2460 JSR MOVE.TEMP1.ARG XP/Q
0963- 4C FF FF 2470 JMP DMULT
2480 *-----
0966- A9 9E 2490 .5 LDA #CON.PI.2 FOR VERY SMALL X
0968- A0 08 2500 LDY /CON.PI.2 SIN(2X/PI) = X*PI/2
096A- 20 FF FF 2510 JSR MOVE.YA.ARG.1
096D- 4C FF FF 2520 JMP DMULT
2530 *-----
2540 * TAN (DAC) = SIN(DAC) / COS(DAC)
2550 *-----
0970- 20 FF FF 2560 DP.TAN JSR PUSH.DAC.STACK SAVE ANGLE
0973- 20 CE 08 2570 JSR DP.SIN TAN=SIN/COS
0976- 20 FF FF 2580 JSR POP.STACK.ARG GET ANGLE
0979- 20 FF FF 2590 JSR PUSH.DAC.STACK SAVE SIN
097C- 20 FF FF 2600 JSR SWAP.ARG.DAC
097F- 20 BF 08 2610 JSR DP.COS GET COSINE
0982- 20 FF FF 2620 JSR POP.STACK.ARG GET SIN
0985- 4C FF FF 2630 JMP DDIV SIN/COS
2640 *-----
0988- 2650 P.ATN .EQ # HART # 5505
03- 2660 P.ATN.N .EQ 3 P3*X^3 + P2*X^2 + P1*X + P0
0988- 42 12 59
098B- 58 02 26
098E- 30 29 54
0991- 72 40 2670 .HS 42.12595.80226.30295.47240 P3
0993- 43 12 55
0996- 79 16 64
0999- 37 98 06
099C- 55 20 2680 .HS 43.12557.91664.37980.65520 P2
099E- 43 29 89
09A1- 28 03 80
09A4- 69 39 62
09A7- 24 48 2690 .HS 43.29892.80380.69396.22448 P1
09A9- 43 19 72
09AC- 03 09 56
09AF- 84 93 50
09B2- 28 54 2700 .HS 43.19720.30956.84935.02854 P0
2710 *-----
09B4- 2720 Q.ATN .EQ #
04- 2730 Q.ATN.N .EQ 4 X^4 + Q3X^3 + Q2X^2 + Q1X + Q0
09B4- 42 37 06
09B7- 60 86 32
09BA- 20 19 02
09BD- 38 01 2740 .HS 42.37066.08632.20190.23801 Q3
09BF- 43 20 76
09C2- 92 68 17
09C5- 33 60 46
09C8- 33 61 2750 .HS 43.20769.26817.33604.63361 Q2
09CA- 43 36 46
09CD- 62 40 32
09D0- 97 70 77
09D3- 62 42 2760 .HS 43.36466.24032.97707.76242 Q1
09D5- 43 19 72
09D8- 03 09 56
09DB- 84 93 50
09DE- 28 c1 2770 .HS 43.19720.30956.84935.02861 Q0
2780 *-----
2790 ATN.TBL.H
09E0- 09 2800 .DA /CON.PI.6
09E1- 08 2810 .DA /CON.PI.2
09E2- 09 2820 .DA /CON.PI.3
2830 ATN.TBL.L
09E3- F1 2840 .DA #CON.PI.6
09E4- 9E 2850 .DA #CON.PI.2
09E5- FC 2860 .DA #CON.PI.3
2870 *-----
09E6- 40 26 79
09E9- 49 19 24
09EC- 31 12 27
09EF- 06 47 2880 CON.TAN.PI.12 .HS 40.26794.91924.31122.70647
09F1- 40 52 35
09F4- 98 77 55
09F7- 98 29 88
09FA- 73 08 2890 CON.PI.6 .HS 40.52359.87755.98298.87308

```

```

09FC- 41 10 47
09FF- 19 75 51
0A02- 19 65 97
0A05- 74 62
0A07- 41 17 32
0A0A- 05 08 07
0A0D- 56 88 77
0A10- 29 35
2900 CON.PI.3 .HS 41.10471.97551.19659.77462
2910 CON.SQR.3 .HS 41.17320.50807.56887.72935
2920 *-----
2930 * ATN FUNCTION
2940 * 1 OR 2 ARGUMENTS
2950 *-----
0A12- 20 B1 00 2960 DP.ATN JSR AS.CHRGET
0A15- 20 B8 DE 2970 JSR AS.CHKOPN CHECK FOR (
0A18- 20 FF FF 2980 JSR DP.EXP GET EXPRESSION
0A1B- 20 FF FF 2990 JSR PUSH.DAC.STACK
0A1E- 20 FF FF 3000 JSR DP.TRUE IN CASE 1 ARGUMENT
0A21- 20 B7 00 3010 JSR AS.CHRGOT
0A24- C9 2C 3020 CMP #1, TWO-ARG?
0A26- D0 06 3030 BNE .1, NO
0A28- 20 B1 00 3040 JSR AS.CHRGET GOBBLE
0A2B- 20 FF FF 3050 JSR DP.EXP YES,GET OTHER ONE
0A2E- 20 FF FF 3060 .1 JSR POP.STACK.ARG GET 1ST ARG BACK
0A31- 20 BB DE 3070 JSR AS.CHKCLS REQUIRE *)
3080 *-----
3090 * ATN (ARG,DAC) ARG/DAC
3100 *-----
3110 DP.ATAN
0A34- AD 0B 08 3120 LDA DAC.SIGN SAVE BOTH SIGNS
0A37- 0A 3130 ASL SIGN OF DENOMINATOR
0A38- AD 17 08 3140 LDA ARG.SIGN SIGN OF NUMERATOR
0A3B- 6A 3150 ROR BIT 7 = DENOM SIGN
0A3C- 8D 19 08 3160 STA UV.SIGN BIT 6 = NUMER SIGN
3170 *---CHECK FOR BOUNDARIES---
0A3F- AD 00 08 3180 LDA DAC.EXPONENT CHECK DENOMINATOR
0A42- F0 0F 3190 BEQ .1 ...V/0, SO RETURN PI/2
0A44- 38 3200 SEC
0A45- AD 0C 08 3210 LDA ARG.EXPONENT
0A48- F0 17 3220 BEQ .12 ...0/U, SO RETURN 0
0A4A- ED 00 08 3230 SBC DAC.EXPONENT
0A4D- 30 0E 3240 BMI .13
0A4F- C9 14 3250 CMP #20 IF >10^20, RETURN PI/2
0A51- 90 14 3260 BCC .11 ...NOT >10^20
0A53- A9 9E 3270 .1 LDA #CON.PI.2 V/0 OR OVERFLOW
0A55- A0 08 3280 LDY /CON.PI.2 SO RETURN PI/2
0A57- 20 FF FF 3290 JSR MOVE.YA.DAC.1
0A5A- 4C 2A 0B 3300 JMP DP.ATN.C
0A5D- C9 C1 3310 .13 CMP #-63 IF <10^-63, RETURN 0
0A5F- B0 06 3320 BCS .11
0A61- 20 FF FF 3330 .12 JSR DP.FALSE RETURN 0
0A64- 4C 1B 0B 3340 .14 JMP DP.ATN.B
0A67- 20 FF FF 3350 .11 JSR DDIV CALCULATE V/U
0A6A- AD 00 08 3360 LDA DAC.EXPONENT
0A6D- C9 36 3370 CMP #340-10 IF X VERY SMALL, ATAN(X)=X
0A6F- 90 F3 3380 BCC .14 ...VERY SMALL INDEED!
3390 *---FOLD AT PI/4---
0A71- A9 00 3400 LDA #0 GET ABS(X), BECAUSE
0A73- 8D 0B 08 3410 STA DAC.SIGN SIGNS ALREADY REMEMBERED
0A76- 8D 18 08 3420 STA N
0A79- AD 00 08 3430 LDA DAC.EXPONENT IS X<1?
0A7C- C9 41 3440 CMP #341
0A7E- 90 0F 3450 BCC .3 ...YES, X<1
0A80- A9 7D 3460 LDA #CON.ONE FORM RECIPROCAL
0A82- A0 08 3470 LDY /CON.ONE
0A84- 20 FF FF 3480 JSR MOVE.YA.ARG.1
0A87- 20 FF FF 3490 JSR DDIV 1/X
0A8A- A9 02 3500 LDA #2 AND REMEMBER WE DID IT
0A8C- 8D 18 08 3510 STA N
3520 *---FOLD AT PI/12---
0A8F- 20 FF FF 3530 .3 JSR MOVE.DAC.TEMP1 SAVE X
0A92- A9 E6 3540 LDA #CON.TAN.PI.12 TAN(PI/12)
0A94- A0 09 3550 LDY /CON.TAN.PI.12
0A96- 20 FF FF 3560 JSR MOVE.YA.ARG.1
0A99- 20 FF FF 3570 JSR DSUB IS X>TAN(PI/12)?
0A9C- AD 0B 08 3580 LDA DAC.SIGN
0A9F- 48 3590 PHA
0AA0- 20 FF FF 3600 JSR MOVE.TEMP1.DAC RESTORE X
0AA3- 68 3610 PLA
0AA4- 10 2F 3620 BPL .4 ...NO, WE DON'T HAVE TO FOLD

```

```

0AA6- EE 18 08 3630      INC N                                ;YES, SO FORM
0AA9- A9 07 3640      LDA #CON.SQR.3          (X*SQR(3)-1) / (SQR(3)+X)
0AAB- A0 0A 3650      LDY /CON.SQR.3
0AAD- 20 FF FF 3660      JSR MOVE.YA.ARG.1
0AB0- 20 FF FF 3670      JSR DMULT                X*SQR(3)
0AB3- 20 FF FF 3680      JSR MOVE.DAC.ARG
0AB6- 20 FF FF 3690      JSR DP.TRUE
0AB9- 20 FF FF 3700      JSR DSUB                X*SQR(3)-1
0ABC- 20 FF FF 3710      JSR MOVE.DAC.TEMP2      SAVE IT
0ABF- 20 FF FF 3720      JSR MOVE.TEMP1.ARG      GET X
0AC2- A9 07 3730      LDA #CON.SQR.3
0AC4- A0 0A 3740      LDY /CON.SQR.3
0AC6- 20 FF FF 3750      JSR MOVE.YA.DAC.1
0AC9- 20 FF FF 3760      JSR DADD                SQR(3)+X
0ACC- 20 FF FF 3770      JSR MOVE.TEMP2.ARG
0ACF- 20 FF FF 3780      JSR DDIV                THE ANSWER
0AD2- 20 FF FF 3790      JSR MOVE.DAC.TEMP1      SAVE FOLDED-UP X
0AD5- 20 FF FF 3800      *---ATAN(0...PI/12)-----
0AD8- 20 FF FF 3810      JSR MOVE.DAC.ARG
0ADB- 20 FF FF 3820      JSR DMULT                X^2
0ADE- A9 88 3830      JSR MOVE.DAC.TEMP2      SAVE X^2
0AE0- A0 09 3840      LDA #P.ATN
0AE2- A2 03 3850      LDY /P.ATN
0AE4- 20 FF FF 3860      LDX #P.ATN.N
0AE7- 20 FF FF 3870      JSR POLY.N
0AEA- A9 B4 3880      JSR MOVE.DAC.TEMP3
0AEC- A0 09 3890      LDA #Q.ATN
0AEE- A2 04 3900      LDY /Q.ATN
0AF0- 20 FF FF 3910      LDX #Q.ATN.N
0AF3- 20 FF FF 3920      JSR POLY.1
0AF6- 20 FF FF 3930      JSR MOVE.TEMP3.ARG      GET P
0AF9- 20 FF FF 3940      JSR DDIV                P/Q
0AFC- 20 FF FF 3950      JSR MOVE.TEMP1.ARG      GET X
0AFF- 20 FF FF 3960      JSR DMULT                P(X^2)/Q(X^2)*X
0AFF- AE 18 08 3970      *---UNFOLD FROM PI/12, PI/4-----
0B02- F0 17 3980      LDX N                                0, 1, 2, OR 3
0B04- CA 4000      BEQ DP.ATN.B                ...NO ADDEND
0B05- F0 08 4010      DEX                                0, 1, OR 2
0B07- AD 0B 08 4020      BEQ .5                ...NO COMPLEMENT
0B0A- 49 80 4030      LDA DAC.SIGN                ATAN(1/X)=ATAN(PI/2 - X)
0B0C- 8D 0B 08 4040      EOR #$80
0B0F- BD E3 09 4050      STA DAC.SIGN
0B12- BC E0 09 4060      LDA ATN.TBL.L,X          GET A(N)
0B15- 20 FF FF 4070      LDY ATN.TBL.H,X
0B18- 20 FF FF 4080      JSR MOVE.YA.ARG.1
0B1B- 2C 19 08 4090      JSR DADD                X + A(N)
0B1E- 10 0A 4100      *---UNFOLD INTO QUADRANTS-----
0B20- A9 A9 4110      DP.ATN.B
0B22- A0 08 4120      BIT UV.SIGN                TEST SIGN OF DENOMINATOR
0B24- 20 FF FF 4130      BPL DP.ATN.C                ...POSITIVE, 1ST OR 4TH
0B27- 20 FF FF 4140      LDA #CON.PI                ...NEGATIVE, 2ND OR 3RD
0B2A- 2C 19 08 4150      LDY /CON.PI                SO DO PI-X
0B2D- 50 08 4160      JSR MOVE.YA.ARG.1
0B2F- AD 0B 08 4170      JSR DSUB
0B32- 49 80 4180      *-----
0B34- 8D 0B 08 4190      DP.ATN.C
0B37- 60 4200      BIT UV.SIGN                TEST SIGN OF NUMERATOR
0B3A- 49 80 4210      BVC .6                ...POSITIVE, 1ST OR 2ND
0B3D- 8D 0B 08 4220      LDA DAC.SIGN                ...NEGATIVE, 3RD OR 4TH
0B3F- 60 4230      EOR #$80                -X
0B42- 4240      STA DAC.SIGN
0B45- 4250      RTS

```

EPROM Programmer

A new EPROM Programmer, called the PROMGRAMER, is out from SCRG (the makers of quikLoader). This one burns anything from 2716's up to 27256's, and retails at \$149.50. We'll sell 'em to you for a nice round \$140. The software comes on disk, with instructions for loading it into EPROM for the quikLoader card.

...also burns 27512's!

More Detail on Using 65C02's in old Apples.....Andrew Jackson

In recent issues of AAL there have been several articles on the 65C02 and how to get it running in the Apple II+. I too was keen to get a 65C02 working in my machine, and had spent some time trying to get first a 1MHz part and then a 2MHz part to work.

William D. O'Ryan's letter in the June 84 AAL prompted me to try again and I am happy to report that the modification he described does work (replacing the LS257's at B6 and B7 with F257's). I wanted to find exactly why I could not simply substitute a 65C02 for a 6502, and so I spent some time looking at the circuit and specifications, using an oscilloscope to check my results.

The reasons that I eventually came up with are as follows. The Apple II circuit relies on various 'features' of the 6502 so that all the various parts of the Apple will work. The circuit diagram shows that the system timing is derived from ϕ_0 ; the 6502 actually expects system timing to be derived from ϕ_2 . There is a slight delay between these two signals: on a 6502 it is about 50ns and on a 65C02 it is about 30ns. This difference in delays is what causes the problems when fitting a 65C02.

To simplify its circuit design the Apple uses a rather dirty trick when reading data from RAM memory. Normally when the 6502 reads data it expects the data on the bus to be valid 100ns before the end of ϕ_2 , and it latches the data into its internal registers when ϕ_2 changes. The setup time allows the data bus to settle into a consistent state before being read. The Apple reduces the setup time to about 45 ns (worst case). This setup time would be ample for the 65C02 were it not for the shift between ϕ_0 and ϕ_2 ; this shift reduces the setup time to 25ns. A 2MHz 65C02 specifies a MINIMUM 40ns setup time; obviously there is a -15ns tolerance on the setup time, and hence the processor works erratically when timings fall into worst case conditions.

The tolerance is regained by substituting 74F257's for the two 74LS257's at board locations B6 and B7. These two chips multiplex the RAM data and the keyboard data; in doing so they add a delay of 30ns worst case to the data. By substituting F257's, the added delay is reduced to 5 ns; this changes the tolerance on the data setup time from -15ns to +10ns.

The Apple //e must use a slightly modified technique when reading data from RAM which explains why a 65C02 works in it without any modifications. I cannot check this as I do not have a //e circuit description. Anyway, it is probably all inside the MMU chip.

[The 65816 specifications state a minimum read data setup time of 50ns, 10ns longer than the 65C02. One AAL reader has called us to report that the 65802 works wonderfully well in his old II+, even better than the original 6502. Some of you have wondered where to get the F257's: try Jameco Electronics, 1355 Shoreway Road, Belmont, CA 94002, phone (415) 592-8097. Their ad in Byte, Dec '84, page 349, says they have 74F257's at \$1.79 each. (editor)]

Gary Little's New Book, "Inside the Apple //"e"

This is a useful book. The kind you want to keep, read, and constantly use as a reference. About 400 pages thick, 6x9, published by Brady Communications at \$19.95.

Gary, a lawyer in Vancouver, has been serious about Apples since 1978 (almost as long as me). He's a long-time subscriber to AAL, Call APPLE, and other sources of the in-depth knowledge crammed into his book. He's also a programmer, with serious software on the market such as "Modem Magician". He knows what he's writing about, and writes it well.

A walk through the chapters may be the quickest way to get the measure of the book.

- 1--condensed history of Apple; intro. to binary, hex, and assembly language.
- 2--inside the 6502 itself: zero page, stack, registers, status, opcodes, address modes, I/O, interrupts, and the memory layout in the //"e.
- 3--the Apple monitor: the commands explained, plus a table of the most useful subroutines in the monitor ROM.
- 4--Applesoft: memory map, tokenization, variable storage, integer and real numbers, the CHRGET subroutine, linking to assembly language programs, subroutines in ROM, and more.
- 5--DOS: internal structure, memory map, page 3 vectors, VTOC, catalog, track/sector lists, RWTS, and a read.sector program. ProDOS: memory map, page 3 vectors, volume bit map, directory, MLI, and a read.block program.
- 6--character input and the keyboard: RDKEY, 80-column firmware, RDCHAR, reading a line, changing input devices, encoding of keys, auto-repeat, type-ahead, all about RESET.
- 7--character and graphic output: too much to list here, all the way through double hi-res.
- 8--memory management: bank switching of ROM and RAM, auxiliary RAM, running co-resident programs.
- 9--speaker and cassette ports: music and voice.
- 10--game port: experiments, push button inputs, annunciators, strobe.
- 11--peripheral slots: I/O memory locations, slot ROM, expansion ROM, scratchpad RAM, auxiliary slot, software protocols.

Many useful and interesting programs are listed in the book. There is an optional diskette available (coupon bound in the book offers it for \$20). The diskette also includes a few bonus utility programs for use with DOS 3.3, including RAMDISK and DISK MAP.



FONT DOWNLOADER & EDITOR (\$39.00)

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed etc.) apply to custom fonts. Full HIRES screen editor lets you create your own characters and special graphics symbols. Compatible with many parallel printer I/F cards. User driver option provided. For Apple II, II+, //e. Specify printer: Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/100, or OkiData 92/93.

NEW !!! The Font Downloader & Editor for the Apple Imagewriter Printer. For use with Apple II, II+, //e (with SuperSerial card) and the new Apple //c (with builtin serial interface).

NEW !!! FONT LIBRARY DISKETTE #1 (\$19.00) contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

DISASM 2.2e - AN INTELLIGENT DISASSEMBLER (\$30.00)

Investigate the inner workings of machine language programs. DISASM converts machine code into meaningful, symbolic source. Creates a standard text file compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor and Pg Zero names included.) An address-based triple cross reference table is provided to screen or printer. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his new ASSEMBLY COOKBOOK. For entire Apple II family including the new Apple //c (with all the new opcodes). **SOURCE CODE** available for an additional \$30.00

S-C Assembler (Ver 4.0 only) SUPPORT UTILITY PACKAGE (\$30.00)

- * SC.XREF - Generates a GLOBAL LABEL Cross Reference Table for complete documentation of source listings.
- * SC.GSR - Global Search & Replace eliminates tedious manual renaming of labels. Search all/part of source.
- * SC.TAB - Tabulates source files into neat, readable form. **SOURCE CODE** available for an additional \$30.00

The 'PERFORMER' CARD (\$39.00)

Plugs into any slot to convert a 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu eliminates need to remember complicated ESC codes. Features include perforation skip, auto page numbering with date & title. Includes large HIRES graphics & text screen dumps. Specify printer: MX-80 with Graftrax-80, MX-100, MX-80/100 with Graftraxplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93. **SOURCE CODE: \$30.00**

FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM (\$25.00)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support. Uses superset of Apple's Comm card and Micromodem II commands. **SOURCE CODE: \$50.00**

RAM/ROM DEVELOPMENT BOARD (\$30.00)

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into \$Cn00-CnFF and \$C800-CFFF.

NEW !!! C-PRINT For The APPLE //c (\$99.00)

Connect standard parallel printers to an Apple //c. C-PRINT is a hardware accessory that plugs into the standard Apple //c printer serial port. The other end plugs into any printer having a standard 36 pin centronics-type parallel connector. Just plug in and print! High speed data transfer at 9600 Baud. No need to reconfigure serial port or load software drivers for text printing.

Avoid a \$3.00 postage/handling charge by enclosing full payment with order. (Mastercard & VISA excluded)

RAK-WARE 41 Ralph Road W. Orange N J 07052 (201) 325-1885



Each chapter ends with a bibliography of related books, manuals, and articles. (You'll find lots of references to AAL.)

If you grew along with Apple, as I did, you probably don't really need this book. On the other hand, you will still enjoy it, and probably want it for you collection. If you are relatively new, and having difficulty gathering all the information from past publications and scattered sources, you will want Gary's book too.

As you might suspect, we like the book so well we have decided to stock it. You can get from us for \$18 plus shipping (and tax where applicable).

Correction re MVN and MVP in 65802.....Bob Sander-Cederlof

In the October AAL I presented a general memory mover written in 65802 code. I stated that the MVP and MVN instructions took 3 cycles-per-byte during the move. I was wrong.

In looking through small tiny print in the preliminary documentation for the chip, I came across the number "7". Shocked, I wrote a little test program which moved 10000 bytes 1000 times. That means the MVN in my test would move a total of 10,000,000 bytes. With a stop watch I clocked the running time at just under 70 seconds. If it had been 3 cycles-per-byte, the test would have run in 30 seconds.

I don't know how I got that "3" in my head, but the right number is "7". Still considerably faster than 6502, though.

```

1000 *SAVE S.TIME MVN
1010      .OP 65816
1020      .OR $300
1030      *-----
00-      1040 CNTR      .EQ 0 AND 1
1050      *-----
1060      MVN.TIMER
1070      CLC          65816 MODE
1080      XCE
1090      REP #$30     16-BIT MODE
1100      *-----
000304- 18      1110      LDA ##1000
000301- FB      1120      STA CNTR
000302- C2 30   1130      *-----
1140      .1      LDX ##$3000 Source start address
00030C- A0 00 40 1150      LDY ##$4000 Destination start address
00030F- A9 0F 27 1160      LDA ##$9999 # Bytes - 1
000312- 54 00 00 1170      MVN 0,0
000315- C6 00   1180      DEC CNTR
1190      *      BNE .1
1200      *-----
000317- 38      1210      SEC          RETURN TO 6502 MODE
000318- FB      1220      XCE
000319- 60      1230      RTS

```

Strange Way to Divide by 7.....Bob Sander-Cederlof

Division by seven is a necessary step for hi-res plotting routines. The quotient is the byte index on a given scan line. The remainder gives the bit position within that byte.

The hi-res code inside the Applesoft ROMs uses a subtraction loop to divide by seven, which can loop up to 36 times at 7 cycles per loop. This is a maximum of over 250 cycles, which is why super-fast hi-res usually uses lookup tables for the quotient and remainder.

I stumbled on a faster way of dividing any value up to 255 by seven. This is not directly usable by standard hi-res, because the x-coordinate can be as large as 279. My trick also does not give the remainder, just the quotient.

Here is the program, along with a test routine which tries every value from 0 to \$FF, printing the quotient. The output from the test program is also shown, and you can see that the quotient is correct in every case. Can you explain why it works?

[Hint: $1/7 = 1/8 + 1/64 + 1/512 + 1/4096 + \dots$]

```

1000 *SAVE S.FUNNY DIVIDE BY SEVEN
1010 *-----
00- 1020 BYTE .EQ 0
1030 *-----
0800- A9 00 1040 T LDA #0
0802- 85 00 1050 STA BYTE
0804- A2 0E 1060 .2 LDX #14
0806- E0 07 1070 .1 CPX #7
0808- D0 05 1080 BNE .4
080A- A9 A0 1090 LDA #$A0
080C- 20 ED FD 1100 JSR $FDED
080F- 20 23 08 1110 .4 JSR DIVIDE.BY.SEVEN
0812- 20 DA FD 1120 JSR $FDDA
0815- E6 00 1130 INC BYTE
0817- F0 09 1140 BEQ .3
0819- CA 1150 DEX
081A- D0 EA 1160 BNE .1
081C- 20 8E FD 1170 JSR $FD8E
081F- 4C 04 08 1180 JMP .2
0822- 60 1190 .3 RTS
1200 *-----
1210 DIVIDE.BY.SEVEN
0823- A5 00 1220 LDA BYTE
0825- 4A 1230 LSR
0826- 4A 1240 LSR
0827- 4A 1250 LSR
0828- 65 00 1260 ADC BYTE
082A- 6A 1270 ROR
082B- 4A 1280 LSR
082C- 4A 1290 LSR
082D- 65 00 1300 ADC BYTE
082F- 6A 1310 ROR
0830- 4A 1320 LSR
0831- 4A 1330 LSR
0832- 60 1340 RTS
1350 *-----
00000000000000 01010101010101
02020202020202 03030303030303
04040404040404 05050505050505
06060606060606 07070707070707
08080808080808 09090909090909
0A0A0A0A0A0A0A 0B0B0B0B0B0B0B
0C0C0C0C0C0C0C 0D0D0D0D0D0D0D
0E0E0E0E0E0E0E 0F0F0F0F0F0F0F
10101010101010 11111111111111
12121212121212 13131313131313
14141414141414 15151515151515
16161616161616 17171717171717
18181818181818 19191919191919
1A1A1A1A1A1A1A 1B1B1B1B1B1B1B
1C1C1C1C1C1C1C 1D1D1D1D1D1D1D
1E1E1E1E1E1E1E 1F1F1F1F1F1F1F
20202020202020 21212121212121
22222222222222 23232323232323
24242424

```

It is possible to divide by 3 or 15 using a program based on the same principle as the divide-by-seven above. Here is the code for those.

```

1210 DIVIDE.BY.FIFTEEN
1220 LDA BYTE
1230 LSR
1240 LSR
1250 LSR
1260 LSR
1270 ADC BYTE
1280 ROR
1290 LSR
1300 LSR
1310 LSR
1320 ADC BYTE
1330 ROR
1340 LSR
1350 LSR
1360 LSR
1370 RTS

```

```

1210 DIVIDE.BY.THREE
1220 LDA BYTE
1230 LSR
1240 LSR
1250 ADC BYTE
1260 ROR
1270 LSR
1280 ADC BYTE
1290 ROR
1300 LSR
1310 ADC BYTE
1320 ROR
1330 LSR
1340 ADC BYTE
1350 ROR
1360 LSR
1370 RTS

```

Using the divide by 15, you could make a divide by ten. First multiply the original number by three (by shifting one bit left and adding), then divide by 15 using the above program, and then by 2 (by shifting one bit right). Since $3X/30 = X/10$, there you have it.

BLANKENSHIP BASIC For the Apple II+, IIe, and IIC

1. WHILE-ENDWHILE and REPEAT-UNTIL loops
2. True IF-THEN-ELSE-ENDIF (Using WHEN)
3. PRINT.USING, FILE, MERGE, RANDOMIZE
4. PRINT and TAB commands work in HIRES
5. 80 columns supported on IIe and IIC
6. Full Editor with AUTO-NUM and RENUM
7. Listings are indented automatically
8. Fast SORT, SEARCH and INSTR\$ commands
9. BOX, BOXFILL, DRAW.USING and SOUND
10. No more CHR\$(4) for DOS commands
11. DEFINE and PERFORM named procedures
12. 99% Upward compatible with Applesoft

**** Introductory Offer \$20 postpaid ****

Money back if not entirely satisfied!

mail check to:
John Blankenship
P.O. BOX 47934
Atlanta GA 30362

DOS 3.3 Only

```

1000 REM sample listing
1010 COMPILE
1011 PRINT " N      SIN(N)"
1015 N= 1
1020 REPEAT
1030   WHEN X < 10 THEN
1040     PRINT " ";N,
1050     PERFORM "INPUT"
1060   ELSE
1070     PRINT N,
1080     PERFORM "OUTPUT"
1090   ENDWHEN
1100   PRINT SIN (N)
1101   N= N + 1
1110 UNTIL N = 20
1120 END

1130 DEFINE "OUTPUT"
1140   REM this is a dummy
1150   REM   procedure
1160   FINISH

1170 DEFINE "INPUT"
1180   REM so is this
1190   FINISH

```

Sly Hex Conversion.....Bob Sander-Cederlof

Have you ever wondered what would happen if you added, in the 6502 decimal mode, values that were not decimal? I have. I also wondered if any of the results might be useful.

For example, what happens if I add 0 to \$0A, in decimal mode? The following little piece of code will tell me:

```
CLC
SED                set decimal mode
LDA #$0A
ADC #0
CLD                clear decimal mode
JMP $FDDA          monitor print byte routine
```

Lo! The \$0A turns into \$10! It makes sense, because of course adding zero does not change anything. But the automatic "decimal adjust" that occurs after the add when the 6502 is in decimal mode detects the "A" nybble, generates a carry to the next nybble, and subtracts \$0A.

It turns out the same process turns \$0B into \$11, \$0C into \$12, and so on up to \$0F into \$15.

That is a useful result! That means that I can convert a hex nybble to BCD byte by merely adding zero when in decimal mode!

A little further experimentation will lead to another useful trick. If I add first \$90 and then \$40, both additions in decimal mode, a value between \$00 and \$0F will be converted to the ASCII code for the digits 0-9 and letter A-F. Believe it or not!

The first addition, of \$90, gives us \$90-\$9F. The automatic "decimal adjust" does nothing to \$90-\$99, and carry will be clear afterwards. If the intermediate result was \$9A-\$9F, the decimal adjust will first generate a nybble carry because the A-F nybble is greater than 9, and reduce that nybble by A. The nybble carry will increment the 9 nybble to A, which gets reduced back to 0 and a byte carry is set. This means we end up with \$90-\$99 with carry clear or \$00-\$05 with carry set.

Adding \$40 in the next step brings the \$90-\$99 up to \$30-\$39 (with carry out of the byte, which we will ignore). The \$00-\$05 will be brought up to \$41-\$45, ASCII codes for A-F. Voila!

Useful, but maybe not the best. It turns out that a more traditional approach is only one byte longer and saves a few cycles. With the value \$00-\$0F in the A-register:

```
    CMP #$0A
    BCC .1                0-9
    ADC #6                convert A-F to $11-16
.1  ADC #$30
```

will convert to ASCII.

```

0800- A2 00 1000 *SAVE S.HEX TO DEC
0802- 8A 1010 T LDX #0
0803- 20 DA FD 1020 .1 TXA
0806- A9 AD 1030 JSR $FDDA
0808- 20 ED FD 1040 LDA #"-
080B- 8A 1050 JSR $FDED
1060 TXA
1070 -----
080C- F8 1080 SED
080D- 18 1090 CLC
080E- 69 00 1110 ADC #0
0810- D8 1120 CLD
1130 -----
0811- 20 DA FD 1140 JSR $FDDA
0814- A9 AD 1150 LDA #"-
0816- 20 ED FD 1160 JSR $FDED
0819- 8A 1170 TXA
1180 -----
081A- F8 1190 SED
081B- 18 1200 CLC
081C- 69 90 1210 ADC #90
081E- 69 40 1220 ADC #40
0820- D8 1230 CLD
1240 -----
0821- 20 DA FD 1250 JSR $FDDA
0824- A9 AD 1260 LDA #"-
0826- 20 ED FD 1270 JSR $FDED
0829- 8A 1280 TXA
1290 -----
082A- C9 0A 1300 CMP #10
082C- 90 02 1310 BCC .2
082E- 69 06 1320 ADC #6
0830- 69 30 1330 ADC #30
1340 -----
0832- 20 DA FD 1350 JSR $FDDA
1360 -----
0835- 20 8E FD 1370 JSR $FD8E
0838- E8 1380 INX
0839- E0 10 1390 CPX #16
083B- 90 C5 1400 BCC .1
083D- 60 1410 RTS

```

Now you can monitor and control the world (or at least your part of it) with a little help from APPLIED ENGINEERING

12 BIT, 16 CHANNEL, PROGRAMMABLE GAIN A/D

- All new 1984 design incorporates the latest in state-of-art I.C. technologies.
- Complete 12 bit A/D converter, with an accuracy of 0.02%!
- 16 single ended channels (single ended means that your signals are measured against the Apple's GND) or 8 differential channels. Most all the signals you will measure are single ended.
- 9 software programmable full scale ranges, any of the 16 channels can have any range at any time. Under program control, you can select any of the following ranges: ± 10 volts, ± 5 V, ± 2.5 V, ± 1.0 V, ± 500 mV, ± 250 mV, ± 100 mV, ± 50 mV, or ± 25 mV.
- Very fast conversion (25 micro seconds).
- Analog input resistance greater than 1,000,000 ohms.
- Laser-trimmed scaling resistors.
- Low power consumption through the use of CMOS devices.
- The user connector has +12 and -12 volts on it so you can power your sensors.
- Only elementary programming is required to use the A/D.
- The entire system is on one standard size plug-in card that fits neatly inside the Apple.
- System includes sample programs on disk.

PRICE \$319

A few applications may include the monitoring of • flow • temperature • humidity • wind speed • wind direction • light intensity • pressure • RPM • soil moisture and many more

8 BIT, 8 CHANNEL A/D

- 8 Channels
- 8 Bit Resolution
- On Board Memory
- Fast Conversion (0.78 ms per channel)
- A/D Process Totally Transparent to Apple (looks like memory)

The APPLIED ENGINEERING A/D BOARD is an 8 bit, 8 channel, memory buffered, data acquisition system. It consists of an 8 bit A/D converter, an 8 channel multiplexer and 8 x 8 random access memory.

The analog to digital conversion takes place on a continuous, channel sequencing basis. Data is automatically transferred to on board memory at the end of each conversion. No A/D converter could be easier to use.

Our A/D board comes standard with 0, 10V full scale inputs. These inputs can be changed by the user to 0, -10V, or -5V, +5V or other ranges as needed. The user connector has +12 and -12 volts on it so you can power your sensors.

- Accuracy: 0.1%
- Input Resistance: 20K Ohms Typ

PRICE \$129.00

SIGNAL CONDITIONER

Our 8 channel signal conditioner is designed for use with both our A/D converters. This board incorporates 8 F.E.T. op-amps, which allow almost any gain or offset. For example: an input signal that varies from 2.00 to 2.15 volts or a signal that varies from 0 to 50 mV can easily be converted to 0-10V output for the A/D.

The signal conditioner's outputs are a high quality 16 pin gold I.C. socket that matches the one on the A/D's so a simple ribbon cable connects the two. The signal conditioner can be powered by your Apple or from an external supply.

FEATURES

- 4.5" square for standard card cage and 4 mounting holes for standard mounting. The signal conditioner does not plug into the Apple, it can be located up to 1/2 mile away from the A/D.
- 22 pin .156 spacing edge card input connector (extra connectors are easily available i.e. Radio Shack).
- Large bread board area.
- Full detailed schematic included.

PRICE \$79.00

DIGITAL INPUT/OUTPUT BOARD

- Provides 8 buffered outputs to a standard 16 pin socket for standard dip ribbon cable connection.
- Power-up reset assures that all outputs are off when your Apple is turned on.
- Features 8 inputs that can be driven from TTL logic or any 5 volt source.
- Your inputs can be anything from high speed logic to simple switches.
- Very simple to program, just PEEK at the data.
- Now, on one card, you can have 8 digital outputs and 8 digital inputs each with its own connector. The super input/output board is your best choice for any control application.

The SUPER INPUT/OUTPUT board manual includes many programs for inputs and outputs. A detailed schematic is included!

Some applications include:

Burglar alarm, direction sensing, use with relays to turn on lights, sound buzzers, start motors, control tape recorders and printers, use with digital joystick.

PRICE \$69.00

Please see our other full page ad in this magazine for information on Applied Engineering's Timemaster Clock Card and other products for the Apple.

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with multi-spacer components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products compatible with Apple II and //e.

Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle three year warranty.

Texas Residents Add 5% Sales Tax
Add \$10.00 if Outside U.S.A.

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214) 492-2027
7 a.m. to 11 p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

Remembering When.....Bob Sander-Cederlof

There is a lot of grumbling going on, or at least so says the media. Supposedly Mac owners are MAD over Apple's \$995 price tag for the 512K upgrade kit. And the fact that new buyers get a lower system price makes them even madder.

If it's true, then I guess the computer "for the rest of us" has found a market with a real-estate or Detroit mentality. Haven't they noticed that prices on virtually all electronic items go down every year? (I always say, "If houses and cars had gone the way electronics has over the last 30 years, we would now be able to buy a 3-bedroom home for two dollars and a nice car for 50 cents. Of course they would both fit on the head of a pin....")

I remember when I bought my Apple, with two rows of 4K RAM chips totalling 8K bytes. Adding another row of 4K chips would have cost me about \$50. The price at that time for one set of 8 16K chips was \$520. Through a special arrangement at Mostek, members of our local club were able to get them for \$150. So to raise my Apple from 8K to 48K cost me \$450. Retail price would have been \$1560, plus tax.

Looking back even further, I found a letter from a Raymond Hoobler to the editor of the Journal of Dentistry, from October 1976. Ray owned an Apple 1, which was populated with 1K RAM chips. He was VERY happy with Apple's promise of an upgrade kit consisting of 4K RAM chips for ONLY \$500!

It will not be too long before the price of 256K RAMs drops. Then we can start grumbling about the price of 4-megabyte upgrade kits. Or, we could rejoice at the blessings of ever improving technology, mass marketing, and understanding wives.

APPLE SOFTWARE ENGINEER

Applied Engineering, a major manufacturer of Apple peripherals, has an opening for a 6502 Machine Language Programmer. Apple experience is required.

(214) 492-2027

Look on page A23 in the Apple Supplement in the back of the December 1984 issue of Byte for an excellent article for the hi-res graphics buff: "Preshift-Table Graphics on Your Apple", by Bill Budge, Gregg Williams, and Rob Moore.

The article presents another of Bill Budge's secrets for fast animation using block graphics. If you want to move a block a few dots left or right, it is time-consuming to shift the 7-bits-in-8 dot images. Older techniques stored pre-shifted sets for each image that might be moved. The neater method described in this article stores a 14x256 byte table of all possible shifts of all possible bytes, and uses a fast lookup technique. I am not going to repeat all that here ... get the article.

The article also included some sample programs that used two other tables: a 192 entry address table for the addresses of each hi-res line, and a 280 entry table for the quotient and remainder of each horizontal position. Both of these tables were originally generated by Applesoft programs, and BSAVED. The example program BLOADED them.

It dawned on me that a machine language program to generate those two tables would take less than half a page of code and be considerably faster than BLOADing pre-generated tables. Furthermore, once the tables were generated, the half-page of code could be overlaid with other programs or data. In a commercial product, this could cut down the boot time significantly.

First I wrote a program to generate the 192 addresses. This was almost a hand-compilation of the Applesoft program in the Byte article, but not quite. (I wrote the comments in near-Basic, as you can see.)

Then I merged into that program the stuff to generate the first 192 quotients and remainders. This is the horizontal dot position divided by 7 (7 dots per byte) to give the byte position on the line and the bit position in that byte.

After the 192 trips through that code, I added a loop to generate the rest of the Q/R pairs, from dot position 192 up to 279.

I timed the program by running it 250 times. All 250 took roughly 3 seconds, which means building the tables once takes about 12 milliseconds. Compare that to loading them from disk, which would take at least a half second.

I haven't tried it yet, but I think the preshift tables which were the meat of the Byte article could also be generated by a machine language program much quicker than BLOADing the same. And since the program only needs to be used once, during initialization, it too could be burned after using.


```

1000 *SAVE S.MAKE HIRES ADDRS
1010 *-----
00- 1020 I .EQ 0
01- 1030 JL .EQ 1
02- 1040 JH .EQ 2
03- 1050 K .EQ 3
04- 1060 Q .EQ 4
05- 1070 R .EQ 5
    1080 *-----
0900- 1090 ADDR1 .EQ $900
09C0- 1100 ADDRH .EQ $9C0
0A80- 1110 QUO.1 .EQ $A80
0B40- 1120 QUO.2 .EQ QUO.1+192
0B98- 1130 REM.1 .EQ QUO.1+280
0C58- 1140 REM.2 .EQ REM.1+192
    1150 *-----
0800- A2 00 1160 BUILD LDX #0      FOR I = 0 TO 191 STEP 1
0802- 86 00 1170 STX I      FOR I = 0 TO $50 STEP $28
0804- 86 01 1180 STX JL      FOR J = 0 TO $0380 STEP $0080
0806- 86 02 1190 STX JH
0808- 86 03 1200 STX K      FOR K = 0 TO $1C STEP $04
080A- 86 04 1210 STX Q      QUOTIENT = 0
080C- 86 05 1220 STX R      REMAINDER = 0
    1230 *---BUILD NEXT HI-RES ADDR-----
080E- A5 00 1240 .1 LDA I
0810- 05 01 1250 ORA JL
0812- 9D 00 09 1260 STA ADDR1,X
0815- A9 20 1270 LDA #$20
0817- 05 02 1280 ORA JH
0819- 05 03 1290 ORA K
081B- 9D C0 09 1300 STA ADDRH,X
    1310 *---SAVE NEXT Q/R PAIR-----
081E- A5 04 1320 LDA Q
0820- 9D 80 0A 1330 STA QUO.1,X
0823- A5 05 1340 LDA R
0825- 9D 98 0B 1350 STA REM.1,X
    1360 *---NEXT K-----
0828- 18 1370 CLC
0829- A5 03 1380 LDA K
082B- 69 04 1390 ADC #4
082D- 85 03 1400 STA K
082F- 49 20 1410 EOR #$20
0831- D0 1B 1420 BNE .2
    1430 *---NEXT J-----
0833- 85 03 1440 STA K
0835- A5 01 1450 LDA JL
0837- 49 80 1460 EOR #$80
0839- 85 01 1470 STA JL
083B- D0 11 1480 BNE .2
083D- E6 02 1490 INC JH
083F- A5 02 1500 LDA JH
0841- 49 04 1510 EOR #4
0843- D0 09 1520 BNE .2
    1530 *---NEXT I-----
0845- 85 02 1540 STA JH
0847- 18 1550 CLC
0848- A5 00 1560 LDA I
084A- 69 28 1570 ADC #$28
084C- 85 00 1580 STA I
    1590 *---BUMP Q/R PAIR-----
084E- E6 05 1600 .2 INC R      R COUNTS 0...6
0850- A5 05 1610 LDA R
0852- 49 07 1620 EOR #7      IF R=7, MAKE 0 AND BUMP Q
0854- D0 04 1630 BNE .3      ...NOT 7 YET
0856- 85 05 1640 STA R      ...R=7, SO MAKE IT 0
0858- E6 04 1650 INC Q      AND BUMP Q
    1660 *---NEXT X-----
085A- E8 1670 .3 INX
085B- E0 C0 1680 CPX #192
085D- 90 AF 1690 BCC .1

```

```

1700 *---NOW FINISH Q/R PAIRS-----
1710 *---BETWEEN 192 AND 279-----
085F- A2 00 1720 LDX #0 FOR X = 0 TO 280-192-1
0861- A5 04 1730 .4 LDA Q
0863- 9D 40 1740 STA QUO.2,X
0868- A5 05 1750 LDA R
0868- 9D 58 1760 STA REM.2,X
086B- E6 05 1770 *---BUMP Q/R PAIR AS BEFORE-----
086D- A5 05 1780 INC R
086F- 49 07 1790 LDA R
0871- D0 04 1800 EOR #7
0873- 85 05 1810 BNE .5
0875- E6 04 1820 STA R
1830 INC Q
1840 *---NEXT X-----
0877- E8 1850 .5 INX
0878- E0 58 1860 CPX #280-192
087A- 90 E5 1870 BCC .4
087C- 60 1880 RTS
1890 *-----

```

Blanken ship's Basic.....Bob Sander-Cederlof

John Blankenship has put together an Applesoft enhancement package, at a mouth-watering price. (See his ad elsewhere in this issue for his \$20 introductory offer.) He sent me a review copy, so I tried it out.

BBASIC is a large chunk of machine language code that sits between HIMEM and the DOS file buffers. It also sits between you and Applesoft, hiding itself behind a facade of new editing and listing features. BBASIC takes control even in direct mode, giving you an EDIT command, structured listings, and the ability to skip out of long catalogs.

In pure BBASIC, line numbers are used only as line numbers, not as destinations for GOTOs or GOSUBs. A built-in RENUM command soon convinces you to live this way and like it. In place of line-number branches, you use alphabetic "names" for subroutines, and WHEN-ELSE-ENDWHEN for logic flow. John has also added WHILE-ENDWHILE, REPEAT-UNTIL, CASE, and other structured looping and branching words.

During execution, a special COMPILE verb creates a table of "names" used in your program. This speeds up execution.

Hires Text generation is built-in, along with some extensions to the hires graphics. Musical tone generation with control over pitch, duration, and timbre is also included. You also get SORT, SEARCH, and PRINT USING.

I am just scratching the surface. I didn't like every feature, but there is plenty left over. Worth a lot more than \$20.

By the way, if John's name sounds familiar, it may be because he is the author of "The Apple House", a book on controlling your home published by Prentice-Hall. John also is a Professor at DeVry Institute.

A Solution to Overlapping DOS Patches.....Paul Lewis
Fairfax, Virginia

I have recently resolved a compatibility problem between two desirable sets of DOS 3.3 patches: the RAMdisk of the 192K Neptune extended memory card, and the DOS Dater that comes with Applied Engineering's Timemaster II. It seems they both want to put patches into the same "unused" spaces inside DOS.

After examining the two patches carefully, I found out which parts of the patches were overlapping. Being unable to find a truly unused area inside DOS, I used the technique on page 7.3 of "Beneath Apple DOS" of placing routines in the "safe" area between DOS and its buffers. This seems to work fine. [Until you try to run some other program that does the same thing, like PLE... (editor)]

The file DATER.OBJ0 contains the DOS.DATER patch that I use. I noticed that the patch could be placed anywhere, since there are no internal references. Using an Applesoft program (part of my HELLO), I move the DOS buffers down far enough to fit this code in, and then BLOAD the patches.

```
100 PRINT CHR$(4)"BRUN AUTO NEPTUNE"  
110 PRINT "PSEUDO DISK INSTALLED"  
120 POKE 40192,128 : REM Lower the buffers  
130 PRINT CHR$(4)"MAXFILES 3"  
140 PRINT "BUFFERS MOVED"  
150 PRINT CHR$(4)"BLOAD DATER.OBJ0,A$9CD0"  
160 POKE 45571,15 : REM Patch file name length  
170 POKE 42883,14  
180 POKE 44085,208 : REM Hook DOS to the DATER code  
190 POKE 44086,156  
200 PRINT "DOS DATER INSTALLED"
```

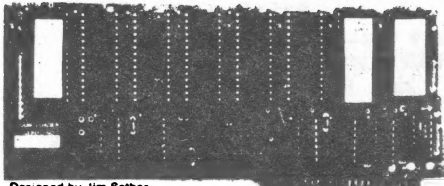
Don Lancaster's AWile TOOLKIT

Solve all of your Applewriter™ IIe hassles with these eight diskette sides crammed full of most-needed goodies including . . .

- Patches for NULL, shortline, IIe detrashing, full expansion
- Invisible and automatic microjustify and proportional space
- Complete, thorough, and fully commented disassembly script
- Detailed source code capturing instructions for custom mods
- Clear and useful answers to hundreds of most-asked questions
- Camera ready print quality secrets (like this ad, ferinstance)
- New and mind-blowing WPL routines you simply won't believe
- Self-Prompting (!) glossaries for Diablo, Epson, many others
- Includes a free "must have" bonus book and helpline service

All this and bunches more for only \$39.95. Everything is unlocked and unprotected. Order from SYNERGETICS, 746 First Street, Box 809-AAL, Thatcher, AZ, 85552. (602) 428-4073. VISA or MC accepted.

quikLoader



Designed by Jim Sather

SPEED

The quikLoader is the *fastest* way to load programs. **BAR NONE!** Applesoft, Integer, or machine language programs can be loaded in fractions of a second. More importantly, DOS is instantly loaded every time the computer is turned on. Integer is even loaded in the language card. This process takes less than a second, saving valuable time. The quikLoader operating system can keep track of over 250 programs stored in PROMs (Programmable Read Only Memory). The user simply transfers any of these programs to PROM using the instructions packed with the unit, and any PROM programmer, or we will provide this service.

CONVENIENCE

How many times have you started to work with a frequently used program, only to find that you have misplaced the disk, or worse, had the disk damaged, or the dreaded "I/O ERROR" message flash on the screen. With the quikLoader, these nightmares can be a thing of the past. Frequently used programs are available *instantly* when you need them, without having to look for the disk, or hoping that the lengthy disk loading procedure goes smoothly. If you do need to use standard disks, the quikLoader even speeds up that process. For example, to catalog a disk, just press ctrl-C Reset. To run the "HELLO" program, press ctrl-H Reset. Other "one-key" commands include entering the monitor, booting the disk, calling up the mini-assembler, etc. The major difference between the quikLoader and the other ROM cards is the complete operating system (in PROM). This enables you to get the quikLoader catalog on the screen (by pressing ctrl-Q Reset), allowing you to see what programs are available. Loading or running of the desired program requires one keypress. Program parameters, such as starting address and length of machine language programs can be seen on the catalog screen, if desired.

EASE OF USE

The quikLoader plugs into any slot of the APPLE][or //e. The card is reset driven. To use any of the many features of the card, RESET is pressed in conjunction with a key. The particular key pressed chooses the feature.

VERSATILE

The quikLoader will accept any of the popular PROMS available on the market, 2716, 2732, 2764, 27128 and 27256. These types may be freely intermixed on the card. Long programs can take up more than one PROM, or several short programs may be stored on one PROM. The quikLoader operating system even handles multiple cards, so you can easily double or triple the amount of PROM memory available. The ultimate memory capacity

of one card is 256K, so many frequently used programs and utilities can be stored. We even start your library of programs with the most popular utilities on the card, FID and COPYA. Now, if you have to copy a disk, you don't have to search for the master disk. You can start copying within 3 seconds after turning on the computer.

INCREASED DISK CAPACITY

Since DOS is loaded from the quikLoader every time the computer is turned on, it is not necessary to take up valuable disk space with DOS. This will give you more than 5% additional space for programs and data on your disks.

SYSTEM REQUIREMENTS

The quikLoader will work in an APPLE][,][+, or //e. If used in a][+, a slightly modified 16K memory card is required in slot 0. A disk drive is required to save data.

\$179.50

DOS, INTEGER BASIC, FID, and COPYA are copyrighted programs of APPLE COMPUTER, INC. licensed to Southern California Research Group to distribute for use only in combination with quikLoader.

NOW AVAILABLE FOR quikLoader:

DOUBLE-TAKE by BEAGLE BROS.

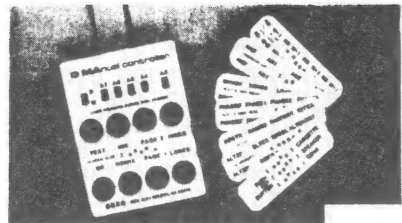
COPY][+ by CENTRAL POINT SOFTWARE

BARKOVITCH I/O TRACER AND SINGLE STEP TRACE

MICRO/TYPOGRAPHER BY TIDBIT SOFTWARE

More programs coming soon.

D Manual controller



Designed by Jim Sather

This hardware product gives the user complete control over all I/O functions in the range \$C0000 through \$C0FFF.

EXAMPLES:

- Switch between TEXT and GRAPHICS.
- Select HIRES or LORES.
- Select Page 1 or 2.
- Turn drives ON or OFF.
- Step head either direction.
- Protect or enable language card.

All this can be done while programs are running. Commands can be issued (via push-buttons) in the middle of a program, and the desired result occurs immediately, without interfering with the normal operation of the program. The card is slot-independent, and is connected to a control panel by a four foot cable.

\$89.95

SCRG PRODUCTS FOR THE APPLE COMPUTER

These items are also available from S-C Software.

quikLoader -- \$170 D Manual Controller -- \$85.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$12 postage for other countries. Back issues are available for \$1.80 each (other countries add \$1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)